PRISM: Physical Reservoir Computing Input Spectral Mapping for Chaotic Time-Series Prediction

Forrest M. Gentry
School of EECS
Oregon State University
Corvallis, Oregon
gentryf@oregonstate.edu

Alex Ochs
School of EECS
Oregon State University
Corvallis, Oregon
ochsal@oregonstate.edu

Sangmin Yoo
School of EECS
Oregon State University
Corvallis, Oregon
sangmin.yoo@oregonstate.edu

***Abstract*—Physical Reservoir Computing (PRC) offers a high-efficiency, low-power alternative to traditional recurrent neural networks by leveraging the intrinsic nonlinear dynamics of physical substrates. However, memristive and other physical reservoir devices often suffer from a limited dynamic range, leading to *Dynamic Saturation* that degrades the reservoir's temporal memory. To address this, we propose Physical Reservoir Computing Input Spectral Mapping (PRISM) that converts time-series dynamics into sets of *evenly sparse* inputs tailored to device-level constraints. PRISM utilizes unevenly-spaced partitioning based on input distribution and soft linear interpolation to maintain information fidelity while ensuring structured sparsity. Our evaluation on canonical chaotic benchmarks (Mackey-Glass, Rossler, Lorenz, and Santa Fe Laser) demonstrates that PRISM improves memory capacity by up to 7× and reduces autonomous forecasting error by up to 33% in 300-step-ahead predictions. These suggest that encoding strategies designed for device-specific limitations are essential for enhancing the performance of physical reservoir systems in complex, real-world temporal tasks.**

***Index Terms*—Reservoir Computing, Memristor, Sparse Coding**

I. Introduction

While modern sequence modelling techniques based on the Transformer architecture have swept the field and set new benchmarks for performance, their computational overhead limits their deployment to low-power environment, especially when compared to Recurrent Neural Networks (RNNs) [1]. Reservoir Computing (RC), a kind of RNNs, does not train its recurrent portion, a *reservoir* of randomly connected nodes that nonlinearly maps temporal information into a higher-dimensional, linearly separable space, which enables a simple linear readout network to perform prediction or classification [2]. Its compact architecture and minimal training cost make RC promising in low-power edge environments.

Physical reservoir computing (PRC) extends this paradigm by implementing the reservoir in physically dynamical substrates, including photonic, spintronic, and memristive devices [3], [4], [5], [6], [7]. These physical hardware platforms offer intrinsic nonlinear state evolution under external stimuli, naturally implementing cumulative information processing and temporally fading memory, both key requirements for RC. As a result, PRC promises low latency and high energy efficiency, making it better-suited to edge computing [6].

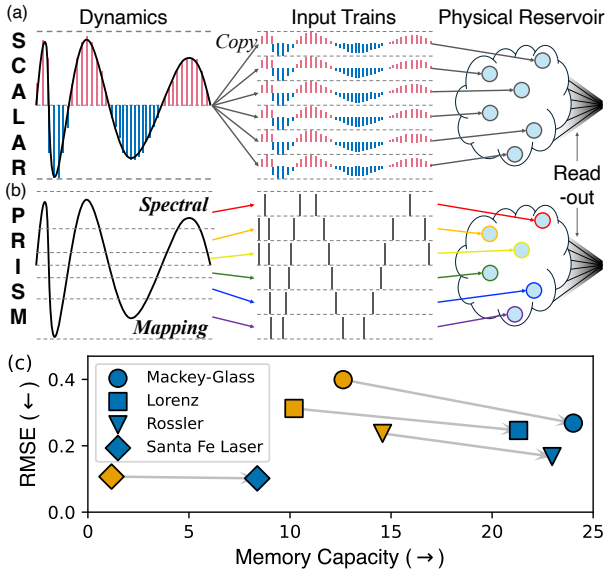

Fig. 1. Illustration of (a) conventional PRC (Scalar) and (b) *PRISM*. While the Scalar method converts time-series input simply into scalar values and pass the identical, dense inputs to all physical reservoir nodes (PRNs), PRISM offers distinct, sparse inputs across PRNs. (c) PRISM (blue) outperforms the Scalar-based PRC (orange) across 4 time-series tasks in prediction error (RMSE) and memory capacity. Arrows on labels represent directions of improvement.

However, physical reservoir devices typically possess a far narrower dynamic range than the nonlinear neurons (e.g.,`tanh`) commonly assumed in RC theory. This challenges manifests as a memory limitation: persistently strong signals can fully saturate the physical reservoir, preventing the system from faithfully encoding and retaining the attractor of interest. This has to be resolved to foster its real-world deployment.

To address the limitation, we propose **Physical Reservoir Computing Input Spectral Mapping (PRISM)** that samples the incoming signal and divides it into evenly sparse spectral "partitions," each mapped to a separate physical reservoir device (Fig 1b). By distributing the input energy across these partitions, PRISM prevents individual physical reservoir nodes from saturating, resulting in quantitative enhancement in memory capacity and performance (Fig 1c) on tasks where such improvements are essential. Our contributions are threefold:

- **Modular PRC Framework.** PRISM converts a time-series dynamics into sparse input patterns tailored to the

dynamical range and decay properties of physical reservoir, minimizing its saturation while keeping nonlinear richness.
- **Improved Memory Capacity by Structured Sparsity.** The input sparsity encourages a physical reservoir to better encode time-series information with minimized loss by the dynamic saturation, which is verified by up to 7× improved memory capacity.
- **Performance Enhancements.** We evaluate the PRISM on four canonical benchmarks (Mackey–Glass, Rossler, Lorenz, Santa Fe Laser), demonstrating consistent improvements in accuracy on autonomous time-series forecasting (up to 33% less error in 300-step ahead prediction).

II. Background and Related Works

A. Reservoir Computing (RC)

RC refers to a class of RNNs where the recurrent part, reservoir, is fixed [2]. The reservoir state evolves following:

$$x_{t+1} = (1-\alpha)x_t + \alpha \tanh(W_{res}x_t + W_{in}u_{t+1}) \quad (1)$$

where $x_t \in \mathbb{R}^N$ is the state of reservoir with *N* nodes, $u_t \in \mathbb{R}^{N_{in}}$ is the input signal, $W_{res} \in \mathbb{R}^{N\times N}$ is the recurrent weight matrix, and $W_{in} \in \mathbb{R}^{N\times N_{in}}$ is the input projection matrix. The leak rate $\alpha \in (0, 1]$ controls the reservoir dynamics timescale.

The final output $\hat{y}_t$ is typically generated via a linear readout:

$$\hat{y}_t = W_{out}[x_t; 1] \quad (2)$$

where W_{out}, is optimized to minimize the error between $\hat{y}_t$ and the ground truth y_t.

B. Physical Reservoir Computing (PRC)

PRC instantiates the reservoir in physical reservoir nodes (PRNs), whose intrinsic device dynamics form the computational substrate. Previous PRC implementations have utilized photonic delay lines, spintronic oscillators, and memristive conductance dynamics, among others [7]. In these systems, the input signal is transduced into a physical quantity (e.g., optical intensity, magnetization, voltage amplitude), which then evolves according to the native physical dynamics.

In particular, the direct injection of the scalar input into PRNs has been adopted for chaotic time-series prediction tasks (Fig 1a). In this *Scalar* method, PRC hardware exhibits nonlinear translation of the input into their internal state dynamics, which beneficially expands the range of feature transformations available to the readout [8]. Also, the internal state natively decays toward its ground state, which offers short-term, *leaky* memory of past information that reservoirs must retain. In specific, memristive reservoirs have exploited the nonlinear behavior and short-term decay of their conductance upon inputs, while simultaneously leveraging device non-idealities to further expand this feature transformation space.

Formally, we model each PRN i with a state variable $x_{i,t}$ whose evolution is governed by an input-driven update rule

$$x_{i,t+1} = x_{i,t} + f(x_{i,t}, u_t; \theta_i) - g(x_{i,t}; \theta_i), \quad (3)$$

where u_t denotes the external input at time t, $f(\cdot)$ captures the device's nonlinear input-driven dynamics (replacing the `tanh` nonlinearity in Eq. 1), and $g(\cdot)$ represents the intrinsic decay dynamics corresponding to the leaky term in Eq. 1. The parameter vector θ_i encodes device-specific variations, which induce heterogeneity across PRNs. The native nonlinearity f and decay g jointly implement the reservoir evolution described by Eq. 1 at a lower cost than conventional hardware (e.g., GPUs), which makes PRC better-suited to low-power environments. The device variation, as captured by θ_i, effectively enriches the feature space and improves representational capacity [3], functioning as a mechanism by which separate representations of an input signal are constructed in parallel.

C. Virtual Nodes

The intrinsic decay dynamics of PRC constrains accurate prediction in complex tasks where long-range temporal relationship matters. To address this, virtual node concept is adopted, whereby the physical reservoir state is sampled and stored at specific temporal offsets within each input interval. This temporal multiplexing effectively extends its usable memory beyond the intrinsic time constants of PRC, which offers long-term temporal information to the readout [9].

III. Problem Formulation: Dynamic Saturation

Despite the potential efficiency and expressiveness of PRC, its performance can be limited under the Scalar input injection schemes by a phenomenon we refer to as *dynamic saturation*. When the full, continuous-valued input is delivered to PRNs at each timestep, they may experience sustained excitation that suppresses their native decay dynamics. This erases the influence of prior inputs on the current state, degrading the reservoir's memory capacity. Importantly, this degradation cannot be remedied by virtual nodes, as the successive samples from the saturated reservoir increases redundancy rather than restoring meaningful memory. Additionally, the saturation heavily restricts the benefits by device variability, as suppressed decay dynamics additionally limits the diversity of expression devices may exhibit. These motivate the need for alternative strategies that preserve the reservoir's intrinsic dynamics.

IV. Methods

To address the dynamic saturation in PRC, we propose *Physical Reservoir Computing Input Spectral Mapping* (PRISM) for input encoding that is designed to limit dense excitation of PRNs. PRISM enforces structured sparsity of input excitation while preserving fine-grain information through two coupled mechanisms: distribution-aware input domain partitioning and locally continuous interpolation between adjacent partitions.

A. Distribution-Aware Structured Sparsity

PRISM regulates sparsity by partitioning the input domain according to the distribution of time-series inputs. Instead of uniform division, the domain is divided into partitions containing approximately equal numbers of input samples, corresponding to statistical quantiles. Each partition is paired with a PRN, and inputs mapped to that partition excite the corresponding PRN. This construction balances the expected

activation frequency across PRNs, yielding approximately uniform sparsity even for highly skewed input distributions.

The partitioning begins by sorting all input samples by value to form the sequence S. The sorted sequence is then divided into D partitions using edges e_i ($i \in \{0, \ldots, D\}$), defined by

$$e_i = S[i \cdot n], \tag{4}$$

where $n = A/D$ denotes the number of samples per partition and A is the total number of input samples. This produces unevenly spaced partitions in the input domain, with narrower intervals in densely populated regions and vice versa (Fig 2).

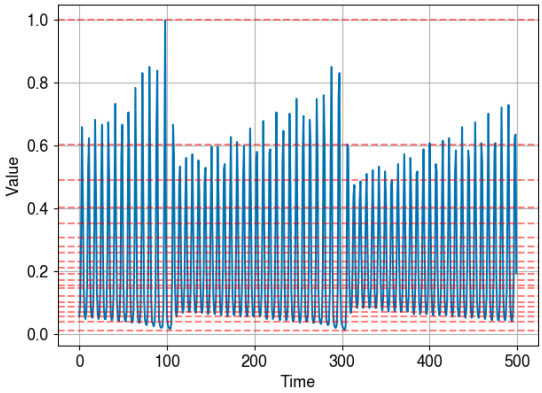

Fig. 2. Illustration of unevenly-spaced edges (red) in the input domain of the Santa Fe Laser time-series task.

By preventing any single partition from being disproportionately activated, this distribution-aware structure limits sustained over-driving of specific PRNs. Consequently, PRISM reduces the likelihood of dynamic saturation and promotes greater temporal diversity in the reservoir state by preserving "breathing room" for intrinsic fading-memory dynamics.

B. Locally Continuous Sparse Encoding

While distribution-aware partitioning regulates sparsity, effective processing of nonlinear and chaotic signals also requires preserving fine-grained information in the input. PRISM therefore employs a locally continuous encoding scheme in which each input value contributes to at most two adjacent partitions. This avoids collapsing all values within a partition to an identical drive signal, ensuring that small changes in the input remain reflected in the encoded representation.

PRISM achieves this by interpolating between neighboring partitions based on the relative position of the input within the enclosing partition interval. This interpolation produces a continuous-valued, locally supported encoding that maintains sensitivity to fine-scale temporal structure while retaining the sparse excitation pattern enforced by the partitioning.

The interpolation mechanism factors in the width of each partition, since they vary in scale. PRISM does this by identifying the two nearest partition edges for a given input, e_i and e_{i+1}. The normalized input, s_t, is then transformed to the selected partition scale according to a min-max normalization,

$$u_t = \frac{s_t - e_i}{e_{i+1} - e_i}, \tag{5}$$

where $u_t \in [0, 1]$ represents a placement within the relative partition, with 0 being the lower edge (e_i) and 1 being the upper edge (e_{i+1}). u_t is then shifted to the center of the partition index to form a continuous partition coordinate,

$$c_t = i + u_t - \tfrac{1}{2}, \tag{6}$$

where the offset ensures that integer-valued coordinates correspond to partition centers rather than edges. From this coordinate, the two nearest partition indices are selected as

$$i_0 = \lfloor c_t \rfloor, \qquad i_1 = i_0 + 1, \tag{7}$$

with indices clipped to the valid range. Linear interpolation weights are then assigned based on proximity,

$$w_1 = c_t - i_0, \qquad w_0 = 1 - w_1, \tag{8}$$

such that $w_0, w_1 \in [0, 1]$ and $w_0 + w_1 = 1$. The input is finally encoded as a sparse pulse vector $p_t \in \mathbb{R}^D$, where only the two adjacent partitions receive nonzero activation,

$$p_t[k] = \begin{cases} w_0, & k = i_0, \\ w_1, & k = i_1, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Each input therefore excites at most two PRNs, yielding a piecewise-linear, locally supported encoding.

Fig 3 illustrates this encoding behavior by comparing with the conventional Scalar scheme. It highlights that PRISM offers more distinct, sparse inputs across PRNs, while Scalar provides identical, dense inputs to PRNs. Accordingly, the physical reservoir shows saturated, heavily correlated PRN states in Scalar while PRISM enables unsaturated, more diverse states.

V. Experiments

Setup: All experiments were performed on an NVIDIA H100 or H200 GPU using PyTorch [10]. Random seeds were fixed (`1337`) for reproducibility. Train and test splits, batch sizes, and optimizer settings were identical across all tasks.

A. Datasets and Data Generation

We evaluate our methods on four canonical time-series benchmarks commonly used in reservoir computing: Mackey–Glass, Lorenz, Rössler, and Santa-Fe laser datasets. The synthetic chaotic systems are generated using standard parameterizations:

- **Mackey–Glass**: discrete-time formulation with parameters $n = 10$, $\tau = 18$, $\beta = 0.2$, and $\gamma = 0.1$, initialized with $x_0 = 1.2$ and history $x_{t<0} = x_0$ [11].
- **Lorenz**: parameters $\rho = 28.0$, $\sigma = 10.0$, and $\beta = 8/3$, corresponding to the canonical chaotic regime [12].
- **Rössler**: parameters $a = 0.2$, $b = 0.2$, and $c = 5.7$, which produce a chaotic attractor [13].

Only the x-component of Rössler and Lorenz are selected to convert the three-dimensional datasets into single dimensional inputs. Santa-Fe laser dataset is obtained from the `ReservoirPy` library [14].

Dataset were min-max normalized and split 8:2 for train/test, with input sequences of length $W + W_{\text{washout}}$ where W is the effective input duration and W_{washout} is the intialization period.

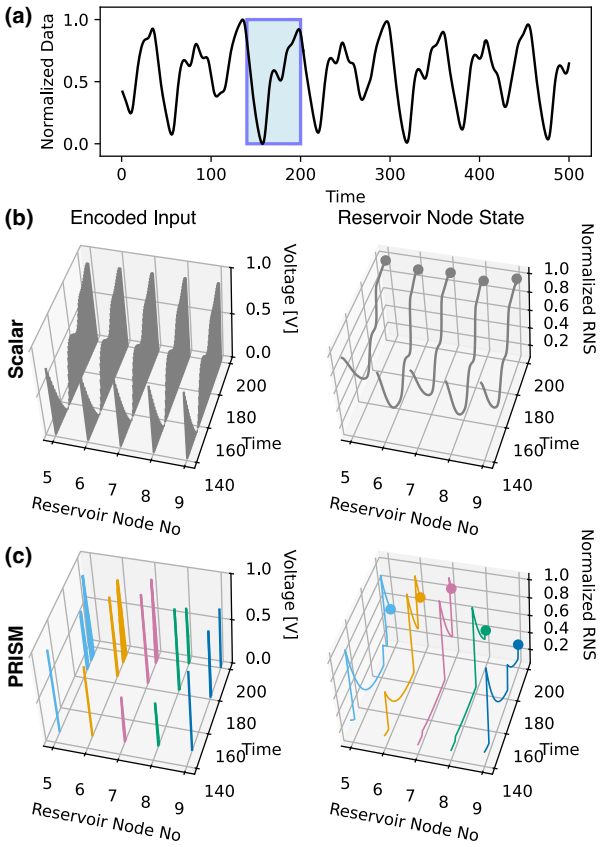

Fig. 3. Comparison among Scalar and PRISM. (a) Input chaotic dynamics (Mackey-Glass). (b-c) Encoded input (left) and the resultant reservoir node states (right) from input dynamics highlighted in blue in (a)). The reservoir node states at time 200 are depicted by dot, which illustrates that PRISM (c) offers much more diverse, unsaturated features than the Scalar (b).

B. Model Overview

PRISM follows a standard PRC architecture composed of three stages: input encoding, a memristor-based physical reservoir, and a linear readout. All non-readout components are fixed during training.

1) Input Encoding: We compare two strategies:

- **Scalar:** The normalized scalar input is applied identically every timestep as a voltage stimulus to all PRN devices.
- **PRISM:** The input signal is encoded, as described in Section IV, producing device-specific pulse trains.

In both cases, the number of physical reservoir nodes (PRNs) is fixed to $D = 20$, matching the number of input partitions.

2) Memristor-based Physical Reservoir: The reservoir consists of an ensemble of D independent WO_x-based RRAM devices, modeled following [3]. Each device evolves according to its conductance state under pulsed voltage excitation:

The conductance update is given by:

$$G_{t+1} = G_t + \Delta G_t, \tag{10}$$

$$\Delta G_t = f(G_t, V_t) - g(G_t), \tag{11}$$

$$f(G_t, V_t) = p_w \eta_1 \sinh(\eta_2 V_t)\, W(G_t, V_t)\, \Delta t, \tag{12}$$

$$g(G_t) = (G_t - G_{\min})(1 - e^{-1/\tau}), \tag{13}$$

where η_1 and η_2 are nonlinear drift parameters, τ is the decay time constant, p_w is the pulse width, and $W(G_t, V_t)$ is a windowing function enforcing physical bounds $[G_{\min}, G_{\max}]$:

$$W(G_t, V_t) = \begin{cases} 1 - \frac{e^{G_t P}}{e^{G_{\max} P}}, & V_t > 0, \\ 1 - \frac{e^{G_{\max} - P(G_t - G_{\min})}}{e^{G_{\max} P}}, & V_t \le 0, \end{cases} \tag{14}$$

with P controlling nonlinearity steepness.

To emulate device variability, we introduce stochasticity:

- Multiplicative pulse noise:

$$\Delta G_{t,\text{noise}} \sim \mathcal{N}(1, \sigma_{\Delta G}) \cdot \Delta G_t, \tag{15}$$

- Per-device decay variation:

$$\tau_n = \tau \cdot \text{clip}(\mathcal{N}(1, \sigma_\tau), [0.5, 1.5]). \tag{16}$$

Each device is simulated for T timesteps, and virtual nodes are obtained by uniformly sampling PRN states at 10 fixed time points over the input window. The resulting reservoir representation is collected into a state tensor

$$\mathbf{H} \in \mathbb{R}^{B \times N \times D}, \tag{17}$$

B and N represent the batch size and the virtual node number.

3) Readout: A linear readout maps the flattened state tensor $x_t \in \mathbb{R}^{B \times (ND)}$ to predictions following Eq. 2.

C. Training Procedure

We follow the standard RC training protocol: only the readout is optimized. We minimize mean squared error,

$$\mathcal{L} = \frac{1}{B} \sum_i (\hat{y}_i - y_i)^2,$$

using the Adam optimizer with learning rate 10^{-3} and batch size $B = 64$. Early stopping is applied based on validation loss, and all evaluations are performed deterministically.

D. Evaluation Protocol

Forecasting performance is evaluated using root mean squared error (RMSE) on one-step-ahead predictions:

$$\text{RMSE} = \sqrt{\frac{1}{T} \sum_t (\hat{y}_t - y_t)^2}.$$

Autonomous prediction is assessed by recursively feeding model outputs back as inputs for up to 300 timesteps. Streaming RMSE over the rollout horizon is used to quantify trajectory stability relative to the ground-truth attractor.

E. Memory Capacity Evaluation

We evaluate reservoir memory using a task specific delayed-input reconstruction metric, referred to as the memory capacity (MC) curve [15]. This probe measures how much past information remains decodable from the current reservoir state. MC is evaluated separately for each dataset and input encoding under matched conditions, with hyper parameters selected based on best autonomous performance. The classical implementation of MC typically utilizes the input of an i.i.d. randomly distributed time series, however, in our implementation we select an input

TABLE I
PERFORMANCE COMPARISON BETWEEN PRISM- AND SCALAR-BASED PRC ON AUTONOMOUS PREDICTION IN RMSE ON 4 DATASETS. LOWER IS BETTER.

	Mackey-Glass		Lorenz		Rössler		Santa Fe Laser	
Steps Ahead	Scalar	**PRISM**	Scalar	**PRISM**	Scalar	**PRISM**	Scalar	**PRISM**
1	0.1185	**0.1225**	0.1203	**0.1256**	0.0756	**0.0646**	0.1656	**0.1611**
84	0.4351	**0.2409**	0.2999	**0.1852**	0.1280	**0.1235**	0.0514	**0.0608**
100	0.4289	**0.2281**	0.3161	**0.2105**	0.1431	**0.1274**	0.0544	**0.0617**
200	0.4218	**0.2365**	0.3222	**0.2473**	0.2024	**0.1641**	0.0772	**0.0735**
300	0.3992	**0.2683**	0.3125	**0.2462**	0.2372	**0.1674**	0.1071	**0.1018**

from the respective time-series task that the model is fit to. This adjustment reflects the behavior of the physical reservoir to respond to input signals, operating within different regimes based on the stimulus. Thus, our metric is useful for comparison between the Scalar and Prism methods, though is not directly comparable to the classical approach.

Given an input sequence u_t and reservoir states $h_t \in \mathbb{R}^D$, we train independent linear readouts to reconstruct delayed inputs u_{t-k} for delays $k = 1, \ldots, K_{\max}$. Reservoir states and targets are mean-centered prior to training. For each delay k, the readout weights w_k are obtained by solving the ridge-regularized least-squares problem

$$w_k = \arg\min_w \ \|X_k w - y_k\|_2^2 + \lambda \|w\|_2^2, \tag{18}$$

where X_k contains reservoir states aligned with delay k, y_k is the corresponding vector of delayed inputs, and $\lambda = 10^{-8}$.

The memory capacity at delay k is defined as the squared Pearson correlation between the reconstructed signal $\hat{y}_k$ and the true delayed input y_k,

$$\mathrm{MC}(k) = \mathrm{corr}^2(\hat{y}_k, y_k). \tag{19}$$

The total linear memory capacity is given by

$$\mathrm{MC}_{\mathrm{sum}} = \sum_{k=1}^{K_{\max}} \mathrm{MC}(k), \tag{20}$$

which summarizes the reservoir's ability to retain linearly decodable information over multiple time delays.

This analysis serves as a diagnostic probe of the reservoir dynamics and does not use the task-trained prediction readout.

VI. RESULTS

Table I reports the root mean squared error (RMSE) obtained using both traditional Scalar input injection and PRISM for autonomous prediction across increasing rollout lengths, ranging from single-step prediction to 300 steps ahead. 84-step ahead prediction result is presented because it was commonly adopted in early RC benchmarks. As the rollout proceeds, prediction accuracy becomes increasingly dependent on the reservoir's ability to retain information about the underlying attractor, rather than on short-term regression accuracy alone.

A. Short-Horizon Behavior

At the single-step prediction horizon, PRISM does not consistently outperform scalar input injection. For Mackey–Glass and Lorenz, the scalar baseline achieves slightly lower RMSE (0.1185 vs. 0.1225 and 0.1203 vs. 0.1256, respectively), while PRISM yields modest improvements for Rössler and Santa Fe Laser. This outcome is expected, as PRISM is designed to influence the reservoir's memory characteristics rather than to optimize instantaneous prediction accuracy. For applications that require only short-horizon prediction, conventional scalar input injection may therefore be sufficient.

B. Memory Capacity Analysis

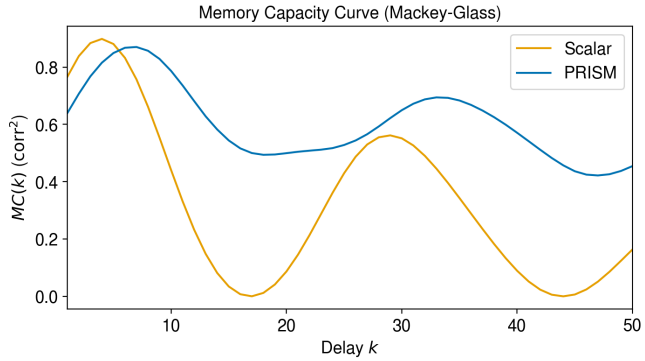

Fig. 4. Sample memory capacity curve for the Mackey–Glass time series. Scalar attains slightly higher MC at short lags, but PRISM maintains consistently higher MC at longer delays.

To directly evaluate the effect of PRISM on the reservoir's temporal retention, we utilize MC for Scalar and PRISM. Fig 4 shows a representative memory capacity curve for Mackey–Glass time series. Scalar injection attains slightly higher memory at very short delays, while PRISM maintains consistently higher memory at longer delays.

PRISM exhibits substantially higher memory capacity across all datasets than Scalar by $1.5 - 7\times$ (Table II). These results indicate that PRISM redistributes MC toward longer effective timescales rather than uniformly increasing capacity at all lags.

TABLE II
AGGREGATE MEMORY CAPACITY ($\mathrm{MC}_{\mathrm{SUM}}$) COMPARISON BETWEEN SCALAR AND PRISM-BASED RCS ACROSS DATASETS. HIGHER IS BETTER.

Methods	Mackey-Glass	Lorenz	Rössler	Santa Fe Laser
Scalar	12.63	10.20	14.58	1.17
PRISM	**24.02**	**21.31**	**22.97**	**8.38**

C. Long-Horizon Prediction

At longer autonomous prediction horizons, the benefits of PRISM become more apparent. While both Scalar and PRISM exhibit increasing error as rollout length grows, PRISM consistently accumulates error at a slower rate for all the datasets. For example, in the Rössler system, Scalar RMSE increases from 0.0756 at one step to 0.2372 at 300 steps, whereas PRISM increases from 0.0646 to 0.1674 over the same interval. Similar trends are observed for Mackey–Glass and Lorenz, where PRISM reduces long-horizon RMSE by on the order of tens of percent relative to Scalar. In comparison, PRISM shows comparable performance in Santa Fe Laser, despite the substantial gains in Memory Capacity. This behavior could be caused by the low-amplitude fluctuations in Santa Fe Laser, which PRISM amplifies due to its non-uniform partitions. Although improvement varies by task, the results indicate that PRISM enhances stability during free-running prediction.

D. Saturation and Memory Hypothesis

Taken together, the observed differences between short- and long-horizon behavior, as well as the variation across datasets and demonstrated improvement in the Memory Capacity metric (table II), are consistent with the hypothesis that frequent saturation of physical reservoir nodes contributes to a degradation of effective reservoir memory, which in turn impacts long-horizon autonomous prediction. PRISM's construction of evenly sparse input partitions reshapes how input energy is distributed across the reservoir and may reduce the likelihood of sustained saturation for signals that strongly excite individual nodes. The observed improvements in long-horizon error accumulation and Memory Capacity supports the interpretation that PRISM primarily influences the reservoir's internal state evolution rather than short-term regression fidelity.

E. Practical Considerations

The distributed partitioning stage introduced by PRISM expands the design space of physical reservoir systems and introduces additional hyperparameters whose optimal values may vary across tasks. Moreover, because the signals considered here are chaotic, finite data samples cannot fully capture long-term variations in dynamic range or power distribution, which may drift over time. Changes in sampling frequency, duration, or preprocessing can therefore alter partition boundaries and affect performance. These considerations highlight the importance of careful parameter selection and suggest that adaptive or task-aware encoding strategies may be required for robust deployment.

VII. Conclusion

Physical reservoir computing remains a promising approach for low-power and edge computing applications, but is constrained by the limited dynamic range and saturation behavior of physical devices. These constraints become particularly apparent in long-horizon autonomous prediction tasks, where performance depends strongly on the reservoir's ability to retain information over time.

In this work, we evaluated PRISM against conventional scalar input injection on four chaotic benchmark datasets. For short-horizon, single-step prediction tasks, performance is comparable between scalar injection and PRISM. At longer prediction horizons, PRISM reduces error accumulation.

These observations are consistent with the hypothesis that structured input sparsification can reduce sustained excitation of individual reservoir nodes and improve effective memory utilization in physical reservoirs. More broadly, the results suggest that input encodings tailored to device-level constraints can improve long-horizon performance without increasing model complexity. Future work will explore adaptive sampling and binning strategies, as well as validation on physical hardware implementations.

References

[1] OpenAI. Gpt-4 technical report, 2024.

[2] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001. The seminal paper on Echo State Networks.

[3] Chao Du, Fuxi Cai, Mohammed Zidan, Wen Ma, Seung Lee, and Wei Lu. Reservoir computing using dynamic memristors for temporal information processing. *Nature Communications*, 8, 12 2017.

[4] Sangmin Yoo, Sieun Chae, Tony Chiang, Matthew Webb, Tao Ma, Hanjong Paik, Yongmo Park, Logan Williams, Kazuki Nomoto, Huili Xing, S. Trolier-McKinstry, Emmanouil Kioupakis, John Heron, and Wei Lu. Efficient data processing using tunable entropy-stabilized oxide memristors. *Nature Electronics*, 7:1–9, 05 2024.

[5] Sieun Chae, Sangmin Yoo, Emmanouil Kioupakis, Wei D. Lu, and John T. Heron. Perspective: Entropy-stabilized oxide memristors. *Applied Physics Letters*, 125(7):070501, August 2024.

[6] Sangmnin Yoo, Eric Yeu-Jer Lee, Ziyu Wang, Xinxin Wang, and Wei D. Lu. Rn-net: Reservoir nodes-enabled neuromorphic vision sensing network. *Advanced Intelligent Systems*, 6(12):2400265, 2024.

[7] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Netw.*, 115(C):100–123, July 2019.

[8] John Moon, Wen Ma, Jong Hoon Shin, Fuxi Cai, Chao Du, Seung Hwan Lee, and Wei D Lu. Temporal data classification and forecasting using a memristor-based reservoir computing system. *Nature Electronics*, 2:480–487, 2019.

[9] Lennert Appeltant, Miguel Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, J. Dambre, Benjamin Schrauwen, Claudio Mirasso, and Ingo Fischer. Information processing using a single dynamical node as complex system. *Nature communications*, 2:468, 09 2011.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. volume 32, pages 8024–8035. Curran Associates, Inc., 2019.

[11] Michael C. Mackey and Leon Glass. Oscillation and Chaos in Physiological Control Systems. *Science*, 197(4300):287–289, July 1977. Publisher: American Association for the Advancement of Science.

[12] Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–148, March 1963.

[13] O. E. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, July 1976.

[14] Nathan Trouvain, Luca Pedrelli, Thanh Trung Dinh, and Xavier Hinaut. ReservoirPy: an Efficient and User-Friendly Library to Design Echo State Networks. In *ICANN 2020 - 29th International Conference on Artificial Neural Networks*, Bratislava, Slovakia, September 2020.

[15] Herbert Jaeger. Short term memory in echo state networks. Technical Report GMD Report 152, German National Research Center for Information Technology (GMD), Neuroinformatics, Bonn, Germany, 2002.